

Utilizing gene regulatory information to speed up the calculation of elementary flux modes

Christian Jungreuthmayer,^{1,2,*} David E. Ruckerbauer,^{1,2} and Jürgen Zanghellini^{1,2,†}

¹*Austrian Centre of Industrial Biotechnology, Vienna, Austria, EU*

²*Department of Biotechnology, University of Natural Resources and Life Sciences, Vienna, Austria, EU*

Despite the significant progress made in recent years, the computation of the complete set of elementary flux modes of large or even genome-scale metabolic networks is still impossible. We introduce a novel approach to speed up the calculation of elementary flux modes by including transcriptional regulatory information into the analysis of metabolic network. Taking into account gene regulation dramatically reduces the solution space and allows the presented algorithm to constantly eliminate biologically infeasible modes at an early stage of the computation procedure. Thereby, the computational costs, such as runtime, memory usage and disk space are considerably reduced. Consequently, using the presented mode elimination algorithm pushes the size of metabolic networks that can be studied by elementary flux modes to new limits.

INTRODUCTION

Elementary flux modes (EFM) are indivisible sets of reactions that represent biologically meaningful pathways [1, 2] under steady state condition. Removing only a single reaction of an EFM results in the extinction of the entire pathway. Consequently, EFMs can be used to decompose metabolic networks mathematically and investigate them unbiasedly. For that reason EFMs have gained increasing attention in the field of metabolic engineering in recent years. However, the computation of EFMs is of combinatorial complexity [3]. Hence, the computational costs for calculating EFMs increase sharply with the size of the analyzed network. The calculation of all EFMs of small networks (up to 50 reactions) is straightforward and simple. However, despite the major progress made recently [4–6] the computation of the complete set of EFMs of large or even genome-scale networks is still impossible. There is a number of tools specifically designed to calculate the complete set of EFMs as performant as possible, such as *Metatool* [7], *CellNetAnalyzer* [8] and *efmtool* [5]. To our best knowledge one of the fastest program currently available is *efmtool* by Marco Terzer which is written in the multi-platform programming language *Java*, supports multi-threading, is published under the open source software license *Simplified BSD Style License* [9], and can be downloaded from [10].

In the presented work we introduce a novel approach to speed up the computation of the complete set of biologically feasible EFMs by taking into account the gene regulatory information of the investigated metabolic network. Transcriptional regulatory networks (TRN) are typically provided as a boolean rule set, e.g. [11]. These rules exclude many of the mathematically possible EFMs for biological reasons. We implemented our algorithm by

extending *efmtool*, thereby, exploiting the full power and advantage of open source software. By utilizing a specific feature of the binary approach [4] which was applied in *efmtool*, the elimination of biologically infeasible modes can be done constantly and at an early stage of the EFM computation process. Thereby, a significant reduction of the computational costs, such as execution time, memory consumption and harddisk space, is achieved.

METHODS

Binary approach

Modern EFM computation programs, such as *efmtool*, use a binary approach [4] of the double description method [12]. In the following we briefly review this binary approach. We will introduce our modifications for the inclusion of transcriptional regulation in the next section.

The binary approach is characterized by splitting each mode into a binary part and a numerical part. The binary part of a mode contains only a single bit for each reaction, where '1' means that the reaction carries a flux and '0' stands for a reaction not carrying a flux. While iterating through the binary algorithm, the numerical part of each mode is successively converted into the binary representation. The iteration procedure terminates when each mode has been completely transformed to its binary form. In a final post-iteration step the computed binary modes are converted back to their numerical forms. The numerical representation of a mode gives the exact stoichiometric amount of each involved reaction that participates in the mode.

We demonstrate the general principals of the binary approach by the simple example network shown in Figure 1. For the sake of clarity the network will not be compressed in order to keep all originally specified reactions and metabolites of the network. In a 'real-life' computation several compression strategies would be applied first

* christian.jungreuthmayer@acib.at

† juergen.zanghellini@acib.at

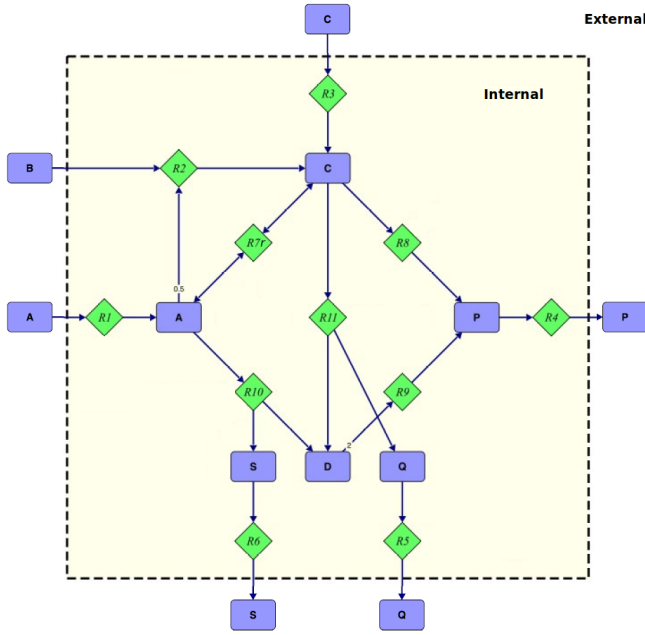


FIG. 1. Example network consisting of 12 metabolites (rectangles) and 11 reactions (diamonds). Only one of the 11 reactions is reversible ($R7r$).

	R1	R2	R3	R4	R5	R6	R7f	R7b	R8	R9	R10	R11
A	1	-0.5	0	0	0	0	-1	1	0	0	-1	0
C	0	1	1	0	0	0	1	-1	-1	0	0	-1
D	0	0	0	0	0	0	0	0	0	-2	1	1
P	0	0	0	-1	0	0	0	0	1	1	0	0
Q	0	0	0	0	-1	0	0	0	0	0	0	1
S	0	0	0	0	0	-1	0	0	0	0	1	0

TABLE I. Extended stoichiometric matrix S_{ext} of the example network shown in Figure 1 after splitting the reversible reaction $R7r$ into the two irreversible reactions $R7f$ and $R7b$.

in order to combine and remove topographically redundant reactions and metabolites [4]. The example network consists of 11 reactions and 12 metabolites. Only reaction $R7r$ is reversible. The stoichiometric matrix S of the example network is shown in Table 1 of the supplementary data section. The external metabolites do not obey the steady state condition and, thus, are irrelevant for the calculation of the EFMs.

First, the reversible reaction $R7r$ is split into a forward and a backward irreversible reaction. This is done by negating the column of the reversible reaction and appending the newly created column right after the original one. Table I shows the extended stoichiometric matrix S_{ext} that only contains irreversible reactions.

The main process of computing all EFMs is based on the double description method [12]. The basic principle of the double description method is to determine an initial set of solution modes which are then iteratively combined and added to the set of existing modes un-

R1	-0.5	0.5	-0.5	0.5	1	0.5
R2	-1	-1	1	1	0	1
R3	1	0	0	0	0	0
R4	0	0	0	1	0.5	0.5
R5	0	0	0	0	0	1
R6	0	0	0	0	1	0
R7f	0	1	0	0	0	0
R7b	0	0	1	0	0	0
R8	0	0	0	1	0	0
R9	0	0	0	0	0.5	0.5
R10	0	0	0	0	1	0
R11	0	0	0	0	0	1

TABLE II. Kernel matrix K of the extended stoichiometric matrix shown in Table I of the example network.

til the complete set of modes is obtained. The solution modes are stored in the mode matrix R that contains one column for each elementary mode. Typically, the initialization of the mode matrix R is obtained by calculating the kernel K of the extended stoichiometric matrix S_{ext} . The kernel K of the extended stoichiometric matrix S_{ext} is defined by $S_{ext}K = 0$ and is shown in Table II.

Next, the initial conversion to the binary representation of the mode matrix R is performed. The final set of elementary modes of the extended network must only contain non-negative flux values, as the extended network contains only irreversible reactions. As pointed out by Gagneur and Klamt [4] using only irreversible reactions is of major importance, as all non-zero elements of a mode will be retained if a new mode is created by combining this mode with other modes that have already been determined during the calculation procedure. Hence, all rows that contain only non-negative values can directly be transformed to the binary representation. For the sake of clarity we use the character f for binary 1 indicating a *flux* carrying reaction and the character n for binary 0 indicating that *no* flux occurs. Usually, the initial solution matrix R is sorted in a way that all rows containing only positive values are at the top. Table III shows the properly sorted mode matrix R containing numerical and binary values before the iteration process is started.

Next, the iteration procedure is performed. Step by step each row that is still in numerical form is transformed to its binary representation. As shown in Table III the next reaction to be processed is $R2$. The double description method requires that all modes containing non-negative values at $R2$ are retained, whereas the modes with negative values are removed. Furthermore, the method requires that all modes with negative values at $R2$ are combined with adjacent modes exhibiting a positive value at $R2$. Hence, the modes M1 and M2 are combined with M3, M4, and M6 resulting in six potential new modes. Two modes are adjacent if the binary part of the new mode is not a superset of any already existing modes - except the two parent modes. For the binary part, the combination of two adjacent

	M1	M2	M3	M4	M5	M6
R3	f	n	n	n	n	n
R4	n	n	n	f	f	f
R5	n	n	n	n	n	f
R6	n	n	n	n	f	n
R7f	n	f	n	n	n	n
R7b	n	n	f	n	n	n
R8	n	n	n	f	n	n
R9	n	n	n	n	f	f
R10	n	n	n	n	f	n
R11	n	n	n	n	n	f
R2	-1	-1	1	1	0	1
R1	-0.5	0.5	-0.5	0.5	1	0.5

TABLE III. Initial mode matrix R. The first ten rows are already transformed to the binary representation where *f* stands for binary 1 and indicates that the reaction carries a flux and *n* stands for binary 0 and indicates that no flux is carried. Note that the order of the reactions has changed compared to Table I in order to maximize the number of rows that can be converted to binary form during the pre-iteration phase.

	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10
R3	n	n	n	n	n	n	n	f	f	f
R4	n	f	f	f	f	f	n	f	f	n
R5	n	n	n	f	f	n	n	f	n	n
R6	n	n	f	n	n	n	n	n	n	n
R7f	n	n	n	n	f	f	f	n	n	n
R7b	f	n	n	n	n	n	f	n	n	f
R8	n	f	n	n	n	f	n	n	f	n
R9	n	n	f	f	f	n	n	f	n	n
R10	n	n	f	n	n	n	n	n	n	n
R11	n	n	n	f	f	n	n	n	f	n
R2	f	f	n	f	n	n	n	n	n	n
R1	-0.5	0.5	1	0.5	0.5	0.5	0.0	0.0	0.0	-0.5

TABLE IV. Mode matrix R after the first iteration step converting reaction *R2* from numerical to binary form.

modes is a simple and fast bitwise OR operation of the involved modes. Combining the numerical part is achieved by a weighted subtraction of the two numerical vectors. The new numerical value v of row r is calculated by $v_{new_r} = (v_{pos_1}v_{neg_r} - v_{neg_1}v_{pos_r})/(v_{pos_1} - v_{neg_1})$, where v_{pos_r} and v_{neg_r} are the values of the positive and of the negative column at row r , respectively. The row index r runs from 1 to n , where row $r = 1$ is the row to be converted at current iteration step and n is the number of rows left to be converted. Applying these instructions to the initial mode matrix R given in Table III results in the new mode matrix shown in Table IV.

Applying the mode combination procedure again for the last row to be converted (*R1*) results in the final mode matrix R as shown in Table V. Now, the matrix R contains only binary elements. Note that the performance of the described iteration procedure for 'real-life' networks can tremendously be increased if tree structures are utilized to store the binary representation of the modes [5].

Next, the futile 2-cycle mode (M6) that was created by splitting the reversible reaction *R7r* is removed. Then the

	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12
R3	n	n	n	n	n	n	f	f	f	n	n	n
R4	f	f	f	f	f	n	f	f	f	f	f	f
R5	n	n	f	f	n	n	f	n	n	f	n	n
R6	n	f	n	n	n	n	n	n	f	n	f	n
R7f	n	n	n	f	f	f	n	n	n	n	n	n
R7b	n	n	n	n	n	f	n	n	f	f	f	f
R8	f	n	n	n	f	n	n	f	n	n	n	f
R9	n	f	f	f	n	n	f	n	f	f	f	n
R10	n	f	n	n	n	n	n	n	f	n	f	n
R11	n	n	f	f	n	n	f	n	n	f	n	n
R2	f	n	f	n	n	n	n	n	n	f	f	f
R1	f	f	f	f	f	n	n	n	n	n	n	n

TABLE V. Mode matrix R after the final iteration step containing only binary values. Mode M6 is the futile 2-cycle mode and can be removed.

	R1	R2	R3	R4	R5	R6	R7r	R8	R9	R10	R11
EFM01	0	1	0	1	0	0	1	1	0	0	0
EFM02	0	1	0	1	1	0	1	0	1	0	1
EFM03	0	1	0	1	0	1	1	0	1	1	0
EFM04	0	0	1	1	0	1	1	0	0	1	0
EFM05	1	1	0	1	0	0	0	1	0	0	0
EFM06	1	1	0	1	1	0	0	0	1	0	1
EFM07	1	0	0	1	0	1	0	0	1	1	0
EFM08	1	0	0	1	1	0	1	0	1	0	1
EFM09	1	0	0	1	0	0	1	1	0	0	0
EFM10	0	0	1	1	1	0	0	0	1	0	1
EFM11	0	0	1	1	0	0	0	1	0	0	0

TABLE VI. Binary representation of all elementary flux modes of the example network shown in Figure 1. 1 means that the reaction carries a flux and 0 means the reaction carries no flux. Note that the futile two-cycle of the reversible reaction *R7r* has already been removed and the forward and backward irreversible reactions (*R7f* and *R7b*) have been combined to the reversible reaction *R7r* by a bitwise OR operation.

irreversible forward and backward reactions *R7f* and *R7b* are combined by a simple bitwise OR operation in order to obtain the reversible reaction *R7r* again. The final set of modes in binary form is shown in Table VI. Reconverting the binary form to the numerical representation is achieved by using the fact that the reduced null space matrix N_{red} multiplied by the sought numerical mode has dimension 1 and is equal to zero. N_{red} is a sub-matrix of the kernel K that only contains columns/reactions where the binary mode to be transformed carries a flux. Hence, only a homogeneous linear system has to be solved to obtain the 1-dimensional solution vector that represents the numerical form of a mode. The result of the reconversion of the binary modes for the simple example network is listed in Table VII.

The binary approach combines several essential advantages: a) modes are stored in binary format which dramatically reduces the memory usage, b) new modes are calculated from existing adjacent modes by using simple

	R1	R2	R3	R4	R5	R6	R7r	R8	R9	R10	R11
EFM01	0.0	1.0	0.0	0.5	0.0	0.0	-0.5	0.5	0.0	0.0	0.0
EFM02	0.0	1.0	0.0	0.25	0.5	0.0	-0.5	0.0	0.25	0.0	0.5
EFM03	0.0	1.0	0.0	0.25	0.0	0.5	-1.0	0.0	0.25	0.5	0.0
EFM04	0.0	0.0	1.0	0.5	0.0	1.0	-1.0	0.0	0.5	1.0	0.0
EFM05	0.5	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
EFM06	0.5	1.0	0.0	0.5	1.0	0.0	0.0	0.0	0.5	0.0	1.0
EFM07	1.0	0.0	0.0	0.5	0.0	1.0	0.0	0.0	0.5	1.0	0.0
EFM08	1.0	0.0	0.0	0.5	1.0	0.0	1.0	0.0	0.5	0.0	1.0
EFM09	1.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0
EFM10	0.0	0.0	1.0	0.5	1.0	0.0	0.0	0.0	0.5	0.0	1.0
EFM11	0.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0

TABLE VII. Numerical representation of all EFM of the example network shown in Figure 1.

bitwise boolean functions which are very fast compared to numeric operations, and c) the bitwise boolean operations used are 'exact', hence, numerical accuracy problems are minimized.

Gene regulation information

TRN control the process of gene expression in cells. They determine how genes activate or repress certain fluxes. Hence, the gene regulatory information of a network imposes additional constraints on the reactions, and, as a consequence, reduces the solution space resulting in a lower number of biologically feasible EFMs. Typically, the gene regulation information is provided in form of boolean functions [11], such as the NOT, OR, and AND operations.

As illustrated in Figure 2 we assume that the gene regulatory network of the example network shown in Figure 1 only consists of a gene *GR* that activates reaction *R7r* and deactivates reactions *R9*. The function of gene *GR* can be transformed to a single boolean expression: $R7r = \text{NOT}(R9)$. This constraint means that the reaction *R7r* must not carry a flux when reaction *R9* carries a flux and vice versa. A simple approach to get the reduced solution space is the application of this gene regulatory rule after all mathematically possible modes were calculated. Naturally, this method does not result in any performance improvement. However, if we consider the basic principle of the binary approach described above, a significant speed up of the computation process can be obtained. The boolean operation $R7r = \text{NOT}(R9)$ implies that the rule is not obeyed if: a) $R9 = 1 = f$ and $R7r = 1 = f$ or b) $R9 = 0 = n$ and $R7r = 0 = n$. The first expression is of particular interest, as it can be used to eliminate all modes from the solution matrix *R* - at any time of the iteration process - if *R9* and *R7r* do carry a flux. This statement is true, as a) the considered mode itself disobeys the rule and b) all children modes generated from the considered mode by combination with other modes

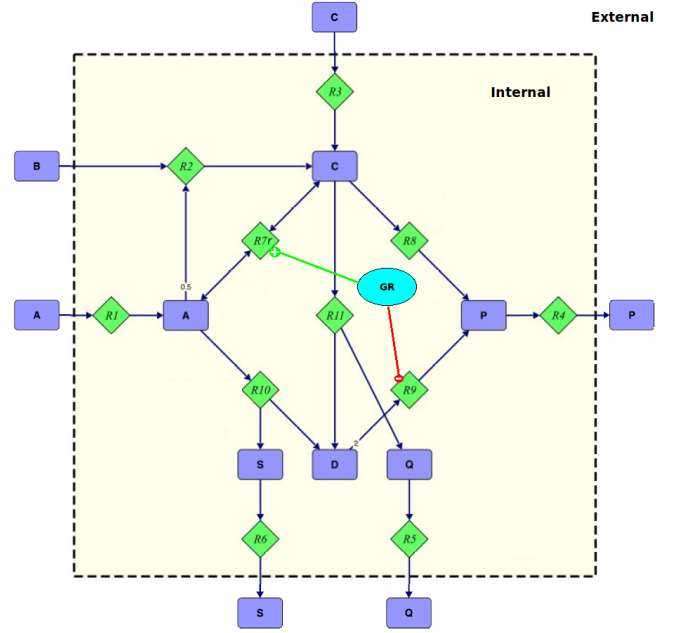


FIG. 2. Example network including the gene regulator network: $R7r = \text{NOT}(R9)$.

will also disobey the rule. The latter property is owed to the fact that a flux value at a certain reaction will be retained by the bitwise OR operation for the rest of the computation procedure (see previous subsection for further details). Removing a mode as soon as possible is of essential importance, as this mode is hindered to create offspring modes that would have to be eliminated at a later stage. The second expression (if $R9 = 0 = n$ and $R7r = 0 = n$) is of no use during the iteration process, as a *no* flux value of *R9* or *R7r* can become a flux carrying reaction in a child mode that is created in a later iteration step. Hence, removing a currently disobeying mode with $R9 = 0$ and $R7r = 0$ would result in the loss of children modes that obey the rule $R7r = \text{NOT}(R9)$. However, the rule $R7r = \text{NOT}(R9)$ for $R9 = 0$ and $R7r = 0$ can still be used to remove infeasible modes after finishing the computation of all binary modes

The above considerations make clear that there are two types of rules: a) rules that can be applied during the iteration process and b) rules that can be applied during the post-processing step after finishing the mode calculation.

Determining if a boolean rule $Ro = \mathcal{B}(R1, \dots, Rn)$ qualifies for the iteration phase is simple. If the output reaction *Ro* of the rule is 0 (does not carry a flux) when all input reactions *R1*, ..., *Rn* are 1 (carry a flux) then the rule can be used during the iteration phase.

Special care must be taken for reversible reactions, as they are split and, hence, occur twice in the extended set of reactions. If either the forward or the backward reaction carries a flux then the original reaction is supposed

	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10
R3	n	n	n	n	n	n	n	f	f	f
R4	n	f	f	f	f	f	n	f	f	n
R5	n	n	n	f	f	n	n	f	n	n
R6	n	n	f	n	n	n	n	n	n	n
R7f	n	n	n	n	f	f	f	n	n	n
R7b	f	n	n	n	n	n	f	n	n	f
R8	n	f	n	n	n	f	n	n	f	n
R9	n	n	f	f	f	n	n	f	n	n
R10	n	n	f	n	n	n	n	n	n	n
R11	n	n	n	f	f	n	n	f	n	n
R2	f	f	n	f	n	n	n	n	n	n
R1	-0.5	0.5	1	0.5	0.5	0.5	0.0	0.0	0.0	-0.5

TABLE VIII. Mode matrix R after the first iteration step. The red font color highlights reactions involved in rule $R7r = \text{NOT}(R9)$ that carry a flux. Mode M5 (highlighted in grey background color) disobeys the rule and is removed from the matrix.

	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11
R3	n	n	n	n	n	f	f	f	n	n	n
R4	f	f	f	f	n	f	f	f	f	f	f
R5	n	n	f	n	n	f	n	n	f	n	n
R6	n	f	n	n	n	n	n	f	n	f	n
R7f	n	n	n	f	f	n	n	n	n	n	n
R7b	n	n	n	n	f	n	n	f	f	f	f
R8	f	n	n	f	n	n	f	n	n	n	f
R9	n	f	f	n	n	f	n	f	f	f	n
R10	n	f	n	n	n	n	n	f	n	f	n
R11	n	n	f	n	n	f	n	n	f	n	n
R2	f	n	f	n	n	n	n	n	f	f	f
R1	f	f	f	f	n	n	n	n	n	n	n

TABLE IX. Mode matrix R after the final iteration step. Mode M8, M9, and M10 do not obey the iteration phase rule. Additionally, mode M1 and M7 disobey the post-processing rule. M5 is also removed, as it is the futile 2-cycle mode created by splitting the reversible reaction $R7r$ into two irreversible reaction.

to be flux carrying when checked against a boolean rule.

Applying these concepts to the example network with the gene regulatory rule $R7r = \text{NOT}(R9)$ results in a mode matrix R after the first iteration step as shown in Table VIII. Table VIII highlights in red font color all reactions involved in rule $R7r = \text{NOT}(R9)$ that carry a flux. It can be seen that mode M5 disobeys the rule and is removed from the matrix R.

In the next step mode M5 does not exist and, hence, fewer adjacency tests have to be performed. Table IX shows the mode matrix R after the final iteration step. It can be seen that mode M8, M9, and M10 do not obey the iteration phase rule, as $R9$ and $R7f$ or $R7b$ carry a flux (highlighted by the red font color). Hence, M8, M9, and M10 can be removed during the iteration phase.

Furthermore, Table IX illustrates that mode M1 and

	R1	R2	R3	R4	R5	R6	R7r	R8	R9	R10	R11
EFM01	0.5	1.0	0.0	0.5	1.0	0.0	0.0	0.0	0.5	0.0	1.0
EFM02	0.0	1.0	0.0	0.5	0.0	0.0	-0.5	0.5	0.0	0.0	0.0
EFM03	1.0	0.0	0.0	0.5	0.0	1.0	0.0	0.0	0.5	1.0	0.0
EFM04	1.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0
EFM05	0.0	0.0	1.0	0.5	1.0	0.0	0.0	0.0	0.5	0.0	1.0

TABLE X. Numerical representation of all EFMs of the example network shown in Figure 1 if the gene regulatory rule $R7r = \text{NOT}(R9)$ is applied.

M7 disobey the post-processing rule, as $R9$, $R7f$, and $R7b$ do not carry a flux value. Consequently, after removing the futile 2-cycle mode M5 the final mode matrix R only contains the five modes M2, M3, M4, M6, and M11. Before transforming the binary modes back to their numerical form the split irreversible reactions $R7f$ and $R7b$ must be combined to the reversible reaction $R7r$. The final set of feasible EFMs is listed in Table X.

Implementation

We implemented our approach as an extension to the open source software *efmtool*. The mode elimination algorithm was realized by adding three *Java* packages to the original version of *efmtool*. The three packages contained ten new *Java* classes. These new classes are responsible for handling the genetic rules and checking the modes against them. Two already existing *Java* classes were slightly enhanced in order to invoke the mode check. The boolean rules are provided by an additional input file using the command line argument **-generule**. The extended version of *efmtool* was compiled by *JDK 1.7.0*. The implementation of the extension was performed as non-invasive as possible, which means that the performance gain might be even better if the new method is integrated to *efmtool* more thoroughly. The mode checks for the iteration phase were implemented using binary bit patterns where the patterns are created simply by setting the involved reactions (all input reactions and the output reaction) to 1 [13]. If a tested mode has every bits set that occurs in the binary bit pattern of a rule then the mode is eliminated. The mode check for the post-processing step was realized by utilizing a reverse polish notation approach that allows a simple and fast execution of boolean functions with any values for the input reactions. The general sequence of operation of our extended version of the binary double description method is shown in the supplementary data section.

RESULTS AND DISCUSSION

We tested our approach on the *E. coli* core model provided by [11, 14]. The model consists of 94 metabolites

Booleanly combined input reactions	Required value of effected output reaction
R_EX_glc.e = 1	→ R_EX2_ac.e = 0
R_EX_glc.e = 1	→ R_ICL = 0
R_EX_glu_L.e = 1	→ R_GLUDy = 0
R_EX_glu_L.e = 1	→ R_GLUSy = 0

TABLE XI. The four boolean rules used by the introduced elimination algorithm to exclude biologically infeasible EFMs during the iteration phase. The 54 other rules can only be applied during the post-processing phase and are shown in the supplementary data section.

and 95 reactions. 59 reactions are reversible. Gene regulatory information for this model is provided by [14] in form of a gene-enzyme-reaction mapping. The mapping was checked for consistency and contradicting rules were removed from the provided data set. The final mapping used by our algorithm to exclude infeasible EFMs contained 58 boolean functions and is shown in Table 2 of the supplementary information section. Only four of these 58 rules qualify for being used during the iteration phase of the EFM computation procedure and are listed in Table XI.

Table XII compares a regular run without regulatory information and a run using the available gene regulation rules. Both runs were performed on a Linux Ubuntu 11.04 computer with 2 Intel Xeon CPUs (6 cores each) and a total of 192 GB of RAM using 10 parallel threads. The table shows that after the iteration phase 226 million modes were obtained without regulatory information. Despite the fact that just four of the 58 boolean rules qualify for the iteration phase, applying the novel mode elimination approach resulted in only 76.7 million modes. This mode removal during the iteration phase caused a reduction of the iteration runtime from 30.9 to 5.3 hours and a decrease of the maximum number of adjacent candidates from $2.2 \cdot 10^{15}$ to $2.6 \cdot 10^{14}$. The elimination of modes in the post-processing phase was even more pronounced, as only 2.1 million of the 76.7 million modes adhered to the boolean gene-enzyme-reaction mapping rules. Interestingly, even the post-processing runtime is reduced by our approach (from 3.2 to 1.8 hours) although 76.7 million binary modes were applied to the post-processing boolean rules. This behaviour is owed to the fact that only 2.6 million modes survive this post-processing filtering procedure which means that only a fraction of the modes were converted from the binary to the numerical representation and were written to disk. This huge reduction of the total number of modes (226.3 million to 2.1 million) had a major influence on the required harddisk space which was decreased from 251 GB to 2.3 GB if an uncompressed double precision text format was used. Table XII clearly shows that considering gene regulatory information in the computation process has a huge impact on the computational key properties

	w/o gene regulation	with gene regulation
No. of modes (iteration)	$226.3 \cdot 10^6$	$76.7 \cdot 10^6$
No. of modes (post-processing)	$226.3 \cdot 10^6$	$2.1 \cdot 10^6$
Max. adjacent candidates	$2.2 \cdot 10^{15}$	$2.6 \cdot 10^{14}$
Max. RAM usage	153 GB	73 GB
Runtime (iteration)	30.9 h	5.3 h
Runtime (post-processing)	3.2 h	1.8 h
Runtime (total)	34.1 h	7.1 h
Disk space	251 GB	2.3 GB

TABLE XII. Comparison of EFM calculation with and without taking into account gene regulatory information. The required disk space is given for a result file containing all modes in text format using double precision. The iteration runtime is the time spent creating the binary modes without pre- and post-processing and the line 'max. adjacent candidates' shows the maximum number of potentially occurring adjacent pairs.

of the calculation of EFMs.

In order to verify that the the presented extension of the *efmtool* computes the correct EFMs, an extra software tool was developed that applies the boolean rules to the complete and unfiltered set of EFMs. The two tools computed identical sets of EFMs ensuring that the *efmtool* extension produces the correct result.

Table XIII shows the development of the number of obtained modes as a function of finished iterations. In total, 21 iteration steps were performed in order to compute the complete set of EFMs. Up to iteration 9 not a single EFM was eliminated and the inclusion of gene regulatory information had no effect. The first removal occurred at iteration 10, where 3 modes were deleted. Although in total only 1.6 million modes were removed during the iteration phase, the final number of modes was reduced by 149.6 million modes. This huge reduction is a result of lost parent modes which otherwise could have spawned a multitude of new children.

In order to find an initial value of the mode matrix R the kernel of the extended stoichiometric matrix is computed. Before the iteration phase is started the reactions are sorted. This is done to put all reactions with only positive values to the top which results in the maximum number of reactions that can be transformed to the binary form before the iteration procedure is even started. Several approaches can be applied to sort the reactions that also contain negative values, e.g. taking no special measures (random order) or ordering by increasing potential combinations (number of negative values times number of positive values). As the iteration phase rules can only be applied if the involved reactions are already converted to the binary representation, it seems beneficial to convert all reactions that are involved in iteration phase rules to the binary form as soon as possible. This concept requires that all these reactions are moved to the top of the set of rows containing negative values. Note that this approach was not implemented in the developed extension but could result in an additional performance gain if realized.

Iteration No.	No. of removed infeasible modes	No. of modes incl. gene regulation	No. of modes w/o gene regulation
9	0	97	97
10	3	205	208
11	0	285	288
12	0	454	457
13	0	456	459
14	0	751	755
15	0	849	952
16	604	2,113	3,223
17	3	6,463	9,454
18	850	17,154	28,114
19	2,168	27,468	48,388
20	0	27,468	48,388
21	1,597	57,180	112,180
22	1,717	244,858	486,847
23	81	224,537	444,371
24	93,933	519,853	1,243,347
25	109,042	1,701,029	4,566,570
26	295,410	2,832,654	8,012,612
27	31,045	4,505,295	12,790,524
28	250,704	12,895,654	37,465,244
29	247,738	26,365,168	77,934,795
30	59,107	32,421,087	94,929,161
31	591,718	76,690,502	226,269,046
sum	1,685,720		

TABLE XIII. Comparison of the number of EFMs as a function of the iteration step for simulations with gene regulatory information and without gene regulatory information.

CONCLUSION

We implemented a novel approach to speed up the computation of the complete set of EFMs of a metabolic network by extending the open source program *efmtool* written by Marco Terzer. Our extension allows the consideration of gene-enzyme-reaction mappings in the process of EFM computation. Biologically infeasible flux modes are constantly eliminated during the calculation process. By implementing an early stage exclusion of modes a considerable reduction of computational costs was achieved which pushes the maximum size of calculable networks to new limits. We think that our approach is another step to the final goal of studying genome-scale metabolic networks by elementary flux modes.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the support by the Federal Ministry of Economy, Family and Youth

(BMWFJ), the Federal Ministry of Traffic, Innovation and Technology (bmvit), the Styrian Business Promotion Agency SFG, the Standortagentur Tirol and ZIT - Technology Agency of the City of Vienna through the COMET-Funding Program managed by the Austrian Research Promotion Agency FFG.

-
- [1] S. Schuster, D. A. Fell, and T. Dandekar, *Nat Biotech*, **18**, 326 (2000), ISSN 1087-0156.
 - [2] S. Schuster, T. Dandekar, and D. A. Fell, *Trends in Biotechnology*, **17**, 53 (1999), ISSN 0167-7799.
 - [3] S. Klamt and J. Stelling, *Molecular Biology Reports*, **29**, 233 (2002), ISSN 0301-4851.
 - [4] J. Gagneur and S. Klamt, *BMC Bioinformatics*, **5**:175 (2004).
 - [5] M. Terzer and J. Stelling, *Bioinformatics*, **24**, 2229 (2008).
 - [6] D. Jevremovića, C. T. Trinh, F. Sreenc, C. P. Sosad, and B. Daniel, *Parallel Computing*, **37**, 261 (2011).
 - [7] A. von Kamp and S. Schuster, *Bioinformatics Application Note*, **22**, 1930 (2006).
 - [8] S. Klamt, J. Saez-Rodriguez, and E. D. Gilles, *BMC Systems Biology*, **1**:2 (2007).
 - [9] Open Source Initiative, “The BSD License,” <http://www.opensource.org/licenses/BSD-2-Clause> (2012).
 - [10] ETH Zurich, Computational Systems Biology Group, “efmtool - Elementary Flux Mode Tool,” <http://www.csb.ethz.ch/tools/efmtool> (2012).
 - [11] J. D. Orth, R. M. T. Fleming, and B. O. Palsson, (2009), doi:10.1128/ecosal.10.2.1.
 - [12] K. Fukuda and A. Prodon, *Combinatorics and Computer Science*, **1120**, 91 (1996).
 - [13] Note that *efmtool* uses an inverse logic where 0 stands for flux carrying reactions and 1 stands for not carrying a flux. Hence, the involved bitwise operations and comparison have to be changed accordingly.
 - [14] University of California, San Diego, Systems Biology Research Group, “The Core E. Coli Model,” <http://gcrg.ucsd.edu/Downloads/EcoliCore> (2012).